Big Data Assignment

ISTD, SUTD

April 7, 2025

Group Assignment

This is a group assignment. You are supposed to work as a team based on your project group. Each group is only required to submit one copy of your solution through eDimension.

Deadline

20 April 2025 23:58.

Synopsis

In this assignment, you are supposed to develop a big data application to process user generated data capturing customer opinions towards brands and businesses.

Your tasks include loading the raw data into the Hadoop distributed file system (HDFS), performing data transformation and cleaning using Spark framework. Finally, you should be able to conduct some descriptive analytics on the cleaned data.

Common Requirements

For all the questions,

1. You may use RDD API and/or Dataframe API. You are not allowed to use Spark SQL API. e.g. `spark.sql("SELECT ....")` will not be rewarded with any mark.
2. The answer should not be dependent on the results from other questions, i.e. for each answer script, it should start from reading the given raw data file. However, you may develop your own library to be shared between answers for different questions.
3. There should be only *one* call to `read()` and *one* call to `write()` in each answer script.

4. Place the input data in the designated HDFS path and output data should be written in the designated HDFS path.

Submission

You are supposed to submit the following

1. `q1.py`
2. `q2.py`
3. `a2.py`
4. `q4.py`
5. `q5.py`
6. `hw2.sh`

Each py file should contain the solution to the correspondent question, i.e. `q1.py` for question 1, `q2.py` for question 2 so on and so forth. The script `hw2.sh` sets up the needed folders and data in HDFS and submits the python scripts to the spark cluster.

Templates of these six files are given to you. Besides editing the python template files, please read through the `hw2.sh` file and update the following sections

```
# change the following according to your student numbers
echo "1001234,1003456"

# change the following according to your environment
hdfs_namenode="localhost"
```

You may assume that during the grading process, the data folder `./data`, the `.py` files and `hw2.sh` file are placed under the same linux folder.

Part 1

Data

In part 1 we are looking at the restaurant review data extracted from

`https://www.kaggle.com/damienbeneschi/krakow-ta-restaurans-data-raw`

You **don't** need to and are **not recommended** to download the data from kaggle. Please make use of the data provided to you along with this assignment.

Copy the **CSV** file `TA_restaurants_curated_cleaned.csv` into HDFS path `/assignment2/part1/input/`. Load it into a Dataframe or RDD, we may observe that the data set has the following schema

```
hdfs_nn = "localhost"
df = spark.read.option("header",True)\
    .csv("hdfs://%s:9000/assignment2/part1/input/" % (hdfs_nn))
df.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- Name: string (nullable = true)
 |-- City: string (nullable = true)
 |-- Cuisine Style: string (nullable = true)
 |-- Ranking: string (nullable = true)
 |-- Rating: string (nullable = true)
 |-- Price Range: string (nullable = true)
 |-- Number of Reviews: string (nullable = true)
 |-- Reviews: string (nullable = true)
 |-- URL_TA: string (nullable = true)
 |-- ID_TA: string (nullable = true)
```

and a preview of the data records looks like the following:

```
+---+--------------------+---------+--------------------+-------+------+-----------+-----------------+--------------------+--------------------+---------+
|_c0|                Name|     City|       Cuisine Style|Ranking|Rating|Price Range|Number of Reviews|             Reviews|              URL_TA|    ID_TA|
+---+--------------------+---------+--------------------+-------+------+-----------+-----------------+--------------------+--------------------+---------+
|  0|Martine of Martin...|Amsterdam|['French', 'Dutch...|    1.0|   5.0|  $$ - $$$|            136.0|[['Just like home...|/Restaurant_Revie...|d11752080|
|  1| De Silveren Spiegel|Amsterdam|['Dutch', 'Europe...|    2.0|   4.5|       $$$$|            812.0|[['Great food and...|/Restaurant_Revie...|  d693419|
|  2|             La Rive|Amsterdam|['Mediterranean',...|    3.0|   4.5|       $$$$|            567.0|[['Satisfaction',...|/Restaurant_Revie...|  d696959|
|  3|            Vinkeles|Amsterdam|['French', 'Europ...|    4.0|   5.0|       $$$$|            564.0|[['True five star...|/Restaurant_Revie...| d1239229|
|  4|Librije's Zusje A...|Amsterdam|['Dutch', 'Europe...|    5.0|   4.5|       $$$$|            316.0|[['Best meal.... ...|/Restaurant_Revie...| d6864170|
|  5|Ciel Bleu Restaurant|Amsterdam|['Contemporary', ...|    6.0|   4.5|       $$$$|            745.0|[['A treat!', 'Wo...|/Restaurant_Revie...|  d696902|
|  6|              Zaza's|Amsterdam|['French', 'Inter...|    7.0|   4.5|  $$ - $$$|           1455.0|[['40th Birthday ...|/Restaurant_Revie...| d1014732|
|  7|Blue Pepper Resta...|Amsterdam|['Asian', 'Indone...|    8.0|   4.5|       $$$$|            675.0|[['Great Experien...|/Restaurant_Revie...|  d697058|
|  8|Teppanyaki Restau...|Amsterdam|['Japanese', 'Asi...|    9.0|   4.5|       $$$$|            923.0|[['Great Food & S...|/Restaurant_Revie...|  d697009|
|  9|Rob Wigboldus Vis...|Amsterdam|['Dutch', 'Seafoo...|   10.0|   4.5|          $|            450.0|[['Excellent Herr...|/Restaurant_Revie...| d1955652|
| 10|      The Happy Bull|Amsterdam|['American', 'Bar...|   11.0|   4.5|  $$ - $$$|            295.0|[['Simply AMAZING...|/Restaurant_Revie...|d10275170|
| 11|             Gartine|Amsterdam|['French', 'Dutch...|   12.0|   4.5|  $$ - $$$|            967.0|[['A hidden gem',...|/Restaurant_Revie...| d1014753|
| 12|     Restaurant Adam|Amsterdam|['French', 'Europ...|   13.0|   4.5|       $$$$|            368.0|[['Love it!', 'As...|/Restaurant_Revie...| d7695005|
| 13|     Biercafe Gollem|Amsterdam|      ['Bar', 'Pub']|   14.0|   4.5|  $$ - $$$|            586.0|[['Awesome little...|/Restaurant_Revie...| d3893242|
| 14|  Restaurant Daalder|Amsterdam|['French', 'Dutch...|   15.0|   4.5|       $$$$|           1246.0|[['Best meal of o...|/Restaurant_Revie...| d1408533|
| 15|Greenwoods Keizer...|Amsterdam|['Dutch', 'Cafe',...|   16.0|   4.5|  $$ - $$$|           1391.0|[['So. Much. Food...|/Restaurant_Revie...| d3200493|
| 16|Omelegg - City Ce...|Amsterdam|['Dutch', 'Europe...|   17.0|   4.5|          $|           1633.0|[['Brunch', 'Wort...|/Restaurant_Revie...| d8562698|
| 17| Brasserie Ambassade|Amsterdam|['French', 'Bar',...|   18.0|   4.5|       $$$$|            958.0|[['Wonderful Chri...|/Restaurant_Revie...| d8567150|
| 18|   Sherpa Restaurant|Amsterdam|['Indian', 'Tibet...|   19.0|   4.5|  $$ - $$$|            426.0|[['Very good tibe...|/Restaurant_Revie...| d6022573|
| 19|La Maschera Lillo...|Amsterdam|['Italian', 'Medi...|   20.0|   4.5|  $$ - $$$|            421.0|[['Fabulous Itali...|/Restaurant_Revie...|d10071792|
+---+--------------------+---------+--------------------+-------+------+-----------+-----------------+--------------------+--------------------+---------+
only showing top 20 rows
```

Question 1 (1 mark)

Develop a Spark application that cleans up the CSV file by removing rows with no reviews or rating < 3.0 as we want to only show positive reviews. Write the output as **CSV** into HDFS path `/assignment2/output/question1/`.

Sample output

```
+---+--------------------+-----+--------------------+-------+------+-----------+-----------------+--------------------+--------------------+---------+
|_c0|                Name| City|       Cuisine Style|Ranking|Rating|Price Range|Number of Reviews|             Reviews|              URL_TA|    ID_TA|
+---+--------------------+-----+--------------------+-------+------+-----------+-----------------+--------------------+--------------------+---------+
|700|Auberge de la Rei...|Paris|[ 'French', 'Euro...|  701.0|   4.0|  $$ - $$$|            489.0|[ [ 'Cozy Restaur...|/Restaurant_Revie...|  d695128|
|701|    Le Petit Vendome|Paris|[ 'French', 'Euro...|  702.0|   4.0|  $$ - $$$|            343.0|[ [ 'Parisian way...|/Restaurant_Revie...| d1146488|
|702|    La Cave Lanrezac|Paris|[ 'Wine Bar', 'Eu...|  703.0|   4.5|  $$ - $$$|            178.0|[ [ 'Dinner with ...|/Restaurant_Revie...|  d812970|
|704|Chez Fernand Chri...|Paris|[ 'French', 'Euro...|  705.0|   4.0|  $$ - $$$|            892.0|[ [ 'Tourist Area...|/Restaurant_Revie...| d1580042|
```

Question 2 (1 mark)

Develop a Spark application that finds the best and the worst restaurants for each city for each price range (in terms of rating). Write the output as **CSV** into HDFS path `/assignment2/output/question2/`. For simplicity, you can ignore rows with `Price Range` field as `null`.

You may use RDD API and/or Dataframe API. You are not allowed to use Spark SQL API.

Sample output:

```
+-----+--------------------+---------+--------------------+-------+------+-----------+-----------------+--------------------+--------------------+---------+
|  _c0|                Name|     City|       Cuisine Style|Ranking|Rating|Price Range|Number of Reviews|             Reviews|              URL_TA|    ID_TA|
+-----+--------------------+---------+--------------------+-------+------+-----------+-----------------+--------------------+--------------------+---------+
| 3198|    Pietersma Snacks|Amsterdam|[ 'Dutch', 'Europ...| 3209.0|   5.0|          $|             null|      [ [  ], [  ] ]|/Restaurant_Revie...|d10587448|
| 2932|     Grillroom Sabba|Amsterdam|[ 'Middle Eastern' ]| 2942.0|   2.5|          $|             12.0|[ [ 'This is a gr...|/Restaurant_Revie...| d6464568|
| 1503|         1 Chefalyon|     Lyon|[ 'Pub', 'Gastrop...| 1485.0|   5.0|       $$$$|             null|      [ [  ], [  ] ]|/Restaurant_Revie...|d12408653|
| 2605|            Papagayo|     Lyon|         [ 'Diner' ]| 2606.0|   2.0|       $$$$|             33.0|      [ [  ], [  ] ]|/Restaurant_Revie...| d1329792|
| 2951|          le bountje| Brussels|[ 'Belgian', 'Eur...| 2952.0|   5.0|   $$ - $$$|             null|      [ [  ], [  ] ]|/Restaurant_Revie...| d1563747|
| 3009|          Belga & Co| Brussels|      [ 'European' ]|   null|  -1.0|   $$ - $$$|             null|      [ [  ], [  ] ]|/Restaurant_Revie...|d13531979|
| 2462|       Sushi Express|Stockholm|[ 'Japanese', 'Su...|   null|   5.0|   $$ - $$$|              2.0|      [ [  ], [  ] ]|/Restaurant_Revie...|d13344590|
```

Question 3 (1 mark)

Develop a Spark application that extracts the two cities with the highest and lowest average rating per restaurant. Combine them, sorted, such that the output looks like this:

For instance:

```
+--------+------------------+-----------+
|    City|     AverageRating|RatingGroup|
+--------+------------------+-----------+
|  Athens| 4.241316931982634|        Top|
|  London| 4.178003263308178|        Top|
.
|Helsinki|3.9153318077803205|     Bottom|
|Brussels| 3.900580875781948|     Bottom|
+--------+------------------+-----------+
```

Write the output as **CSV** files into HDFS path `/assignment2/output/question3/`.

Question 4 (1 mark)

Develop a Spark application that counts the number of restaurants by city and cuisine style.

The output should something like the following:

```
+----------+-------------------+-----+
|      City|            Cuisine|count|
+----------+-------------------+-----+
| Amsterdam|         Vietnamese|   24|
|Bratislava|          Hungarian|    3|
|  Brussels|      International|   74|
|    London|             Kosher|   26|
|      Lyon|      Mediterranean|   80|
|      Lyon|             German|    2|
```

Write the output as **CSV** files into HDFS path `/assignment2/output/question4/`.

Part 2

In this second part of the assignment, we investigate the movie credit data extracted from

`https://www.kaggle.com/tmdb/tmdb-movie-metadata?select=tmdb_5000_credits.csv`

You don't need to and are not recommended to download the data from kaggle. Please make use of the data provided to you along with this part.

For part 2, instead of CSV, we consider the input file in **Parquet** format. Parquet format is a compressed column based format which is optimized for parallel processing. For more details of parquet file format, refer to the following documentation:

`https://spark.apache.org/docs/latest/sql-data-sources-parquet.html`

Copy the `tmdb_5000_credits.parquet` file into HDFS path `/part2/input/`. Load it into a Dataframe or RDD, we may observe that the data set has the following schema

```
df = spark.read.option("header",True)\
    .parquet("hdfs://%s:9000/assignment2/part2/input/" % (hdfs_nn))
df.printSchema()
```

```
root
 |-- movie_id: long (nullable = true)
 |-- title: string (nullable = true)
 |-- cast: string (nullable = true)
 |-- crew: string (nullable = true)
```

If we take a look at the first rows of the data, we see the following:

```
+--------+--------------------+--------------------+--------------------+
|movie_id|               title|                cast|                crew|
+--------+--------------------+--------------------+--------------------+
|   19995|              Avatar|[{"cast_id": 242,...|[{"credit_id": "5...|
|     285|Pirates of the Ca...|[{"cast_id": 4, "...|[{"credit_id": "5...|
|  206647|             Spectre|[{"cast_id": 1, "...|[{"credit_id": "5...|
|   49026|The Dark Knight R...|[{"cast_id": 2, "...|[{"credit_id": "5...|
|   49529|         John Carter|[{"cast_id": 5, "...|[{"credit_id": "5...|
|     559|        Spider-Man 3|[{"cast_id": 30, ...|[{"credit_id": "5...|
|   38757|             Tangled|[{"cast_id": 34, ...|[{"credit_id": "5...|
|   99861|Avengers: Age of ...|[{"cast_id": 76, ...|[{"credit_id": "5...|
|     767|Harry Potter and ...|[{"cast_id": 3, "...|[{"credit_id": "5...|
```

Question 5 (2 marks)

Develop a Spark application that finds the pairs of actors/actresses that are co-cast for **at least** 2 movies. The output should be in a (set of) **Parquet** files in the following schema:

```
movie_id, title, actor1, actor2
```

Note that the result should not contain any repetition, e.g.

```
49026, The Dark Knight Rises, Michael Caine, Christian Bale
```

is considered as a duplicate entry of

```
49026, The Dark Knight Rises, Christian Bale, Michael Caine
```

The output should be something like the following:

```
+--------+--------------------+--------------------+--------------------+
|movie_id|               title|              actor1|              actor2|
+--------+--------------------+--------------------+--------------------+
|   69848|      One Man's Hero|        James Gammon|        Tom Berenger|
|    9942|        Major League|        James Gammon|        Tom Berenger|
|     285|Pirates of the Ca...|        David Bailie|         Ho-Kwan Tse|
|      58|Pirates of the Ca...|        David Bailie|         Ho-Kwan Tse|
|     921|      Cinderella Man|     Michael Stevens|Conrad Bergschneider|
|   14577|          Dirty Work|     Michael Stevens|Conrad Bergschneider|
|   16290|          Jackass 3D|Dimitry Elyashkevich|          Manny Puig|
|   12094|  Jackass Number Two|Dimitry Elyashkevich|          Manny Puig|
|    9012|  Jackass: The Movie|Dimitry Elyashkevich|          Manny Puig|
```

Hint

You should be able to extract the needed info from the `movie_id`, `title` and `cast` columns.

Good luck!